Locally Linear Embedding: Manifold Learning for Nonlinear Dimensionality Reduction

Ahmed BADI
ahmedbadi905@gmail.com
linkedin.com/in/badi-ahmed

January 2026

Abstract

Locally Linear Embedding (LLE) is a popular manifold learning algorithm for nonlinear dimensionality reduction. It assumes that high-dimensional data lie on a smooth, low-dimensional manifold, and that each point can be approximated by a linear combination of its nearest neighbors. LLE first computes local reconstruction weights in the original space, then finds a low-dimensional embedding that preserves these local linear relationships. This article introduces the intuition behind LLE, describes the algorithm step by step, derives its main optimization problems, and discusses practical aspects such as the choice of neighborhood size and dimensionality. We also compare LLE to other dimensionality reduction techniques like PCA, t-SNE, and UMAP, and highlight its advantages and limitations in real-world applications.

Keywords: Locally Linear Embedding, Manifold Learning, Nonlinear Dimensionality Reduction, Neighborhood Graph, Embedding.

1 Introduction

Many real-world datasets are high-dimensional but have underlying low-dimensional structure. For example, images of a rotating object, handwritten digits, or pose sequences often lie on a smooth manifold embedded in a higher-dimensional space. Manifold learning aims to discover this latent low-dimensional structure. [1], [2]

Locally Linear Embedding (LLE), introduced by Roweis and Saul (2000), is a classic manifold learning technique. It belongs to the family of nonlinear dimensionality reduction methods and focuses on preserving local geometric relationships: each point is reconstructed from its nearest neighbors using linear weights, and the same weights are used to reconstruct points in a low-dimensional space. [1], [3]

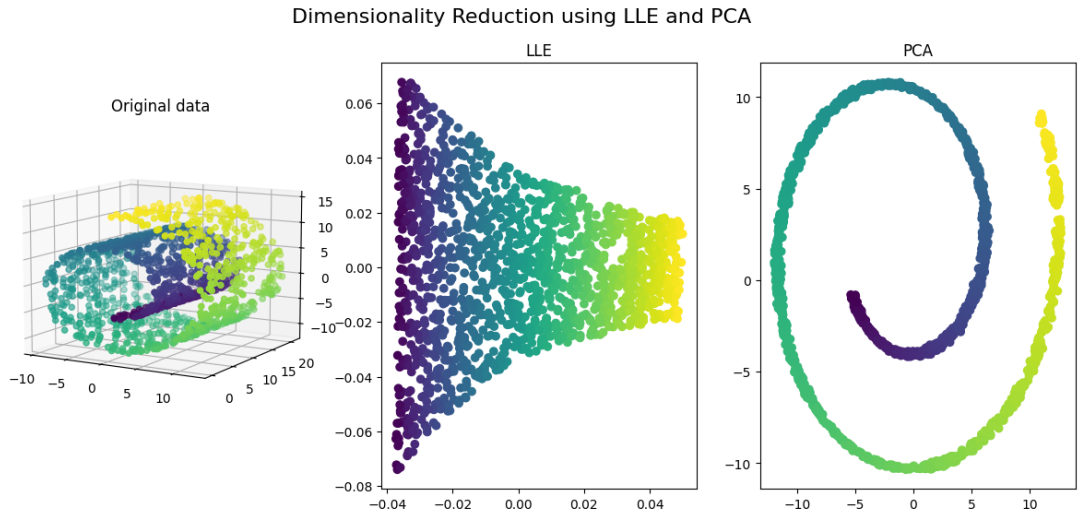

Figure 1: Example: LLE unfolds a 3D Swiss roll dataset into a 2D manifold while preserving local neighborhoods.

LLE is widely used for visualization and exploratory data analysis when the data manifold is expected to be smooth and well-sampled. [4], [5]

2 Basic Idea of LLE

LLE is based on two key assumptions: [1], [3]

- The data points lie on or near a smooth low-dimensional manifold embedded in a high-dimensional space.
- In a small neighborhood, the manifold can be approximated as locally linear.

Given these assumptions, LLE proceeds in two main stages:

1. **Local reconstruction:** For each data point, find a linear combination of its nearest neighbors that best reconstructs it (in the original space).
2. **Global embedding:** Find low-dimensional coordinates such that each point can be reconstructed from its neighbors using the same linear weights.

By preserving local linear reconstruction weights, LLE preserves the manifold's local geometry in the low-dimensional embedding. [4], [6]

3 Problem Setup and Notation

Let $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be data points in $\mathbb{R}^D$, assumed to lie on a d-dimensional manifold ($d \ll D$). LLE aims to find an embedding $\{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ in $\mathbb{R}^d$ that preserves local structures. [1]

We choose:

- k: number of nearest neighbors for each point.
- $W = (W_{ij}) \in \mathbb{R}^{N \times N}$: reconstruction weight matrix, where W_{ij} is non-zero only if $\mathbf{x}_j$ is a neighbor of $\mathbf{x}_i$.

The algorithm minimizes two related cost functions:

- Local reconstruction error in the original space.
- Embedding reconstruction error in the low-dimensional space.

4 Step 1: Neighborhood Selection

For each point $\mathbf{x}_i$, LLE identifies its k nearest neighbors (by Euclidean distance, or other metrics). [6], [7]

Let $\mathcal{N}(i)$ be the index set of neighbors of $\mathbf{x}_i$. The neighborhood graph encodes which points are considered locally related.

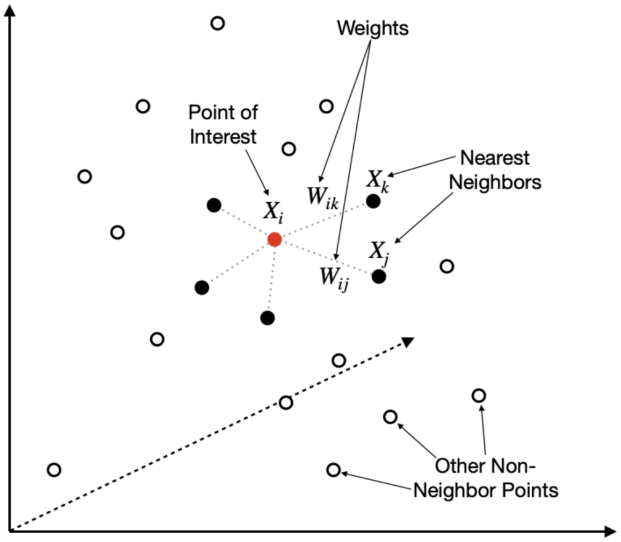

Figure 2: Neighborhood graph for LLE: each point is connected to its k nearest neighbors.

The choice of k is important:

- Too small: neighborhoods are noisy and may disconnect the manifold.
- Too large: neighborhoods may violate local linearity and mix distant regions.

5 Step 2: Reconstruction Weights in High Dimension

For each point $\mathbf{x}_i$, LLE finds weights W_{ij} that reconstruct $\mathbf{x}_i$ from its neighbors: [1], [3]

$$\mathbf{x}_i \approx \sum_{j \in \mathcal{N}(i)} W_{ij} \mathbf{x}_j.$$

The reconstruction error is

$$\mathcal{E}(W) = \sum_{i=1}^{N} \left\| \mathbf{x}_i - \sum_{j \in \mathcal{N}(i)} W_{ij} \mathbf{x}_j \right\|^2.$$

To avoid trivial solutions and make the problem well-posed, LLE constrains the weights:

$$\sum_{j \in \mathcal{N}(i)} W_{ij} = 1 \quad \text{for all } i,$$

and $W_{ij} = 0$ if $j \notin \mathcal{N}(i)$.

For each point $\mathbf{x}_i$, the weights are obtained by solving a constrained least-squares problem:

$$\min_{\{W_{ij}\}} \left\| \mathbf{x}_i - \sum_{j \in \mathcal{N}(i)} W_{ij} \mathbf{x}_j \right\|^2 \quad \text{s.t.} \quad \sum_{j \in \mathcal{N}(i)} W_{ij} = 1.$$

In matrix form, for each i: [3], [4]

- Construct a local covariance matrix $C^{(i)}$ of size $k \times k$:

$$C^{(i)}_{mn} = (\mathbf{x}_i - \mathbf{x}_{j_m})^\top (\mathbf{x}_i - \mathbf{x}_{j_n}), \quad j_m, j_n \in \mathcal{N}(i).$$

- Solve

$$C^{(i)} \mathbf{w}^{(i)} = \mathbf{1},$$

where $\mathbf{w}^{(i)}$ are the unnormalized weights and $\mathbf{1}$ is a vector of ones, then normalize:

$$W_{ij_m} = \frac{w^{(i)}_m}{\sum_n w^{(i)}_n}.$$

Regularization is often used if $C^{(i)}$ is nearly singular (e.g., adding a small multiple of the identity matrix). [5]

6 Step 3: Low-Dimensional Embedding

Once the weights W_{ij} are fixed, LLE finds low-dimensional points $\mathbf{y}_i \in \mathbb{R}^d$ that are reconstructed from their neighbors using the same weights: [1]

$$\mathbf{y}_i \approx \sum_{j \in \mathcal{N}(i)} W_{ij} \mathbf{y}_j.$$

The embedding is obtained by minimizing the cost

$$\Phi(Y) = \sum_{i=1}^{N} \left\| \mathbf{y}_i - \sum_j W_{ij} \mathbf{y}_j \right\|^2,$$

subject to constraints that remove trivial solutions (e.g., centering and fixing scale). Here $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_N]^\top \in \mathbb{R}^{N \times d}$.

6.1 Matrix Formulation

Let I be the $N \times N$ identity matrix. Define the matrix

$$M = (I - W)^\top (I - W).$$

Then the cost can be written as

$$\Phi(Y) = \sum_{i,j} M_{ij} \mathbf{y}_i^\top \mathbf{y}_j = \mathrm{Tr}(Y^\top M Y),$$

where Tr is the trace. [1], [3]

To avoid trivial solutions ($\mathbf{y}_i = \mathbf{0}$ for all i), LLE imposes constraints such as:

$$\frac{1}{N} \sum_{i=1}^{N} \mathbf{y}_i = \mathbf{0}, \quad \frac{1}{N} Y^\top Y = I_d,$$

which fix the origin and scale of the embedding.

Under these constraints, the minimizer Y is given by the bottom $d+1$ eigenvectors of M (excluding the smallest eigenvalue corresponding to the constant vector). The coordinates in $\mathbb{R}^d$ are taken from the eigenvectors associated with the second to $(d+1)$-th smallest eigenvalues. [1], [3]

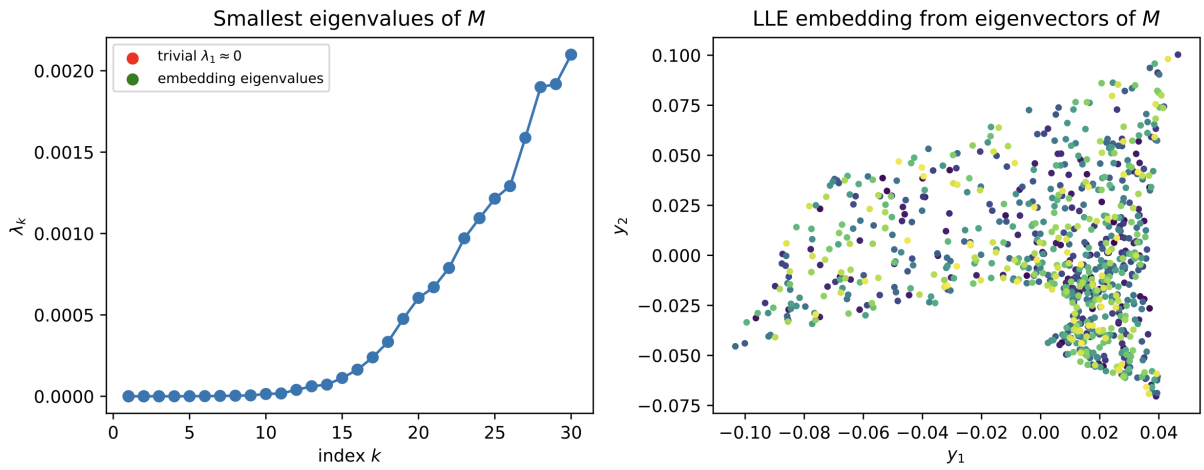

Figure 3: Example of eigenvalue spectrum of the LLE matrix M: the smallest eigenvalue corresponds to a trivial solution, the next d give the embedding.

7 Algorithm Summary

The LLE algorithm can be summarized as follows. [6], [7], [8]

1. **Input:** Data $\{\mathbf{x}_i\}_{i=1}^N \subset \mathbb{R}^D$, neighborhood size k, target dimension d.
2. **Neighbor search:** For each $\mathbf{x}_i$, find the k nearest neighbors $\mathcal{N}(i)$.
3. **Compute weights:** For each i, solve the constrained least-squares problem to obtain reconstruction weights W_{ij} that minimize $\|\mathbf{x}_i - \sum_{j\in\mathcal{N}(i)} W_{ij}\mathbf{x}_j\|^2$, with $\sum_{j\in\mathcal{N}(i)} W_{ij} = 1$.
4. **Construct matrix M:** Set $M = (I - W)^\top(I - W)$.
5. **Eigen-decomposition:** Compute the bottom $d+1$ eigenvectors of M. Discard the smallest (trivial) one and take the next d as coordinates of the embedding.
6. **Output:** Low-dimensional points $\{\mathbf{y}_i\}_{i=1}^N \subset \mathbb{R}^d$.

In practice, highly optimized implementations (e.g., in scikit-learn) handle nearest neighbor search and sparse linear algebra efficiently. [5], [9]

8 Complexity and Practical Considerations

The main computational steps are: [5]

- **Nearest neighbor search**: typically $O(N \log N)$ or more, depending on algorithm and dimension.
- **Weight computation**: solving a $k \times k$ linear system for each point, roughly $O(Nk^3)$.
- **Eigen-decomposition**: computing the bottom $d + 1$ eigenvectors of an $N \times N$ sparse matrix M, using iterative methods.

Choosing hyperparameters:

- Neighborhood size k: often between 5 and 20; should be large enough to capture local structure but small enough to maintain local linearity.

- Target dimension d: typically 2 or 3 for visualization, or slightly larger for downstream tasks.

LLE assumes:

- The manifold is smooth and well-sampled (enough neighbors around each point).
- Local neighborhoods are approximately linear.

If these assumptions are violated (e.g., noisy or sparse data), the embedding may be unstable or distorted. [4], [10]

9 Variants of LLE

Many variants have been proposed to improve stability, scalability, and robustness. [11]

9.1 Modified LLE (MLLE)

Modified LLE (MLLE) introduces multiple local weight vectors for each neighborhood to improve stability when the local least-squares problem is ill-conditioned. It uses multiple linearly independent weight vectors that are approximately optimal, leading to more stable embeddings, especially for higher intrinsic dimensions. [5], [12]

9.2 Other Extensions

Other LLE-based methods include: [11]

- **Kernel LLE**: combines kernel methods with LLE ideas.
- **Supervised and semi-supervised LLE**: incorporate label information.
- **Incremental / Landmark LLE**: handle streaming or very large datasets using landmarks and Nyström approximations. [13]
- **Robust LLE**: address noise and outliers via robust loss functions.

10 Comparison with Other Methods

Table 1 compares LLE with PCA, Isomap, and t-SNE / UMAP at a high level. [1], [2], [14]

Method	**Main idea**	**Structure preserved**	**Typical use**
PCA	Linear projection maximizing variance	Global linear structure	Baseline reduction
Isomap	Geodesic distances on neighborhood graph	Global manifold geometry	Nonlinear manifolds with geodesic structure
LLE	Preserve local linear reconstruction weights	Local neighborhood geometry (topology)	Manifold learning, visualization
t-SNE / UMAP	Match local probability / fuzzy neighbor structure	Local clusters and neighborhoods	2D/3D visualization, cluster exploration

Table 1: Comparison of LLE with other dimensionality reduction methods.

LLE focuses on preserving local linear relationships rather than pairwise distances or global variance, which makes it especially suitable for manifolds where local geometry carries the most important information. [1], [14]

11 Applications

LLE is used in many applications where data lie on nonlinear manifolds: [1], [2]

- **Visualization** of high-dimensional data (images, digits, poses) in 2D or 3D.
- **Pattern recognition** and clustering in an embedded space.
- **Preprocessing** for downstream tasks, where the manifold is more linear in the embedded space.

12 Conclusion

Locally Linear Embedding is a foundational algorithm in manifold learning and nonlinear dimensionality reduction. Its main ideas are simple but powerful:

- Approximate each point as a linear combination of its nearest neighbors in the original space.
- Find a low-dimensional embedding that preserves these reconstruction weights.

LLE often produces meaningful low-dimensional representations for data lying on smooth manifolds and is especially useful for visualization and exploratory analysis. At the same time, it can be sensitive to noise, parameter choices, and sampling density, which has motivated many variants and improvements. [1], [11]

References

[1] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[2] Wikipedia contributors, *Nonlinear dimensionality reduction*, `https://en.wikipedia.org/wiki/Nonlinear_dimensionality_reduction`, 2024.

[3] L. K. Saul and S. T. Roweis, *An introduction to locally linear embedding*, `http://graphics.stanford.edu/courses/cs233-25-spring/ReferencedPapers/lleintro.pdf`, 2002.

[4] Paperspace Blog, *Tutorial: Dimension reduction using lle*, `https://blog.paperspace.com/dimension-reduction-with-lle/`, 2020.

[5] scikit-learn developers, *Manifold learning*, `https://scikit-learn.org/stable/modules/manifold.html`, 2010.

[6] Activeloop, *What is locally linear embedding?* `https://www.activeloop.ai/resources/glossary/locally-linear-embedding-lle/`, 2020.

[7] GeeksforGeeks, *Locally linear embedding in machine learning*, `https://www.geeksforgeeks.org/machine-learning/locally-linear-embedding-in-machine-learning/`, 2023.

[8] CodeSignal, *Exploring locally linear embedding: A dimensionality reduction technique*, `https://codesignal.com/learn/courses/non-linear-dimensionality-reduction-techniques/lessons/exploring-locally-linear-embedding-a`, 2024.

[9] scikit-learn developers, *Locallylinearembedding*, https://scikit-learn.org/stable/modules/generated/sklearn.manifold.LocallyLinearEmbedding.html, 2010.

[10] K. Bryan and T. Leise, "A note on the locally linear embedding algorithm," *SIAM Journal on Matrix Analysis and Applications*, 2006, https://cs.adelaide.edu.au/~wojtek/papers/lle.pdf.

[11] R. Hosseini et al., "Locally linear embedding and its variants," *arXiv preprint arXiv:2011.10925*, 2020.

[12] Z. Zhang and H. Zha, "Modified locally linear embedding using multiple weights," *Advances in Neural Information Processing Systems*, vol. 20, 2007.

[13] M. Schuo et al., *Truly incremental locally linear embedding*, https://ai.stanford.edu/~schuon/learning/inclle.pdf, 2010.

[14] A. Kukushkin, *Manifold learning and dimensionality reduction: Algorithms, applications, pros, and cons*, https://neptune.ai/blog/dimensionality-reduction, 2025.